PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

links between heap objects SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used: **links between heap objects**

Found **108,067** of **207,474**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Prefetch injection based on hardware monitoring and object metadata
Ali-Reza Adl-Tabatabai, Richard L. Hudson, Mauricio J. Serrano, Sreenivas Subramoney
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation PLDI '04**, Volume 39 Issue 6
Publisher: ACM Press
Full text available: pdf(288.00 KB) Additional Information: full citation, abstract, references, citings, index terms

Cache miss stalls hurt performance because of the large gap between memory and processor speeds - for example, the popular server benchmark SPEC JBB2000 spends 45% of its cycles stalled waiting for memory requests on the Itanium® 2 processor. Traversing linked data structures causes a large portion of these stalls. Prefetching for linked data structures remains a major challenge because serial data dependencies between elements in a linked data structure preclude the timely materialization ...

Keywords: cache misses, compiler optimization, garbage collection, prefetching, profile-guided optimization, virtual machines

2 Heap analysis: Recursive data structure profiling
Easwaran Raman, David I. August
June 2005 **Proceedings of the 2005 workshop on Memory system performance MSP '05**
Publisher: ACM Press
Full text available: pdf(173.82 KB) Additional Information: full citation, abstract, references, index terms

As the processor-memory performance gap increases, so does the need for aggressive data structure optimizations to reduce memory access latencies. Such optimizations require a better understanding of the memory behavior of programs. We propose a profiling technique called *Recursive Data Structure Profiling* to help better understand the memory access behavior of programs that use recursive data structures (RDS) such as lists, trees, etc. An RDS profile captures the runtime behavior of the ...

Keywords: RDS, dynamic shape graph, list linearization, memory profiling, shape profiling

3 Is it a tree, a DAG, or a cyclic graph? A shape analysis for heap-directed pointers in

C
Rakesh Ghiya, Laurie J. Hendren
January 1996 **Proceedings of the 23rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '96**
Publisher: ACM Press
Full text available: pdf(1.51 MB) Additional Information: full citation, references, citings, index terms

4 Runtime integrity checking for inter-object connections
Guilin Chen, Mahmut Kandemir
May 2005 **Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design ICCAD '05**
Publisher: IEEE Computer Society
Full text available: pdf(222.31 KB) Additional Information: full citation, abstract

Ensuring integrity of heap resident data is critical for many embedded systems. Ever-scaling process technology combined with power-saving techniques employed in embedded systems is increasing vulnerability of such systems to hardware-related errors such as soft errors. While such errors are transient and do not harm the architecture, they can corrupt data. In this study, we explore solutions to the inter-object connectivity problem in heap memory of Java-based embedded environments. Our objecti ...

5 Ownership confinement ensures representation independence for object-oriented programs
Anindya Banerjee, David A. Naumann
November 2005 **Journal of the ACM (JACM)**, Volume 52 Issue 6
Publisher: ACM Press
Full text available: pdf(664.37 KB) Additional Information: full citation, abstract, references, index terms, review

Representation independence formally characterizes the encapsulation provided by language constructs for data abstraction and justifies reasoning by simulation. Representation independence has been shown for a variety of languages and constructs but not for shared references to mutable state; indeed it fails in general for such languages. This article formulates representation independence for classes, in an imperative, object-oriented language with pointers, subclassing and dynamic dispatch, cl ...

Keywords: Alias control, confinement, data refinement, relational parametricity, simulation

6 HeapMD: identifying heap-based bugs using anomaly detection
Trishul M. Chilimbi, Vinod Ganapathy
October 2006 **ACM SIGARCH Computer Architecture News , ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , Proceedings of the 12th international conference on Architectural support for programming languages and operating systems ASPLOS-XII**, Volume 34 , 40 , 41 Issue 5 , 5 , 11
Publisher: ACM Press
Full text available: pdf(246.89 KB) Additional Information: full citation, abstract, references, index terms

We present the design, implementation, and evaluation of HeapMD, a dynamic analysis tool that finds heap-based bugs using anomaly detection. HeapMD is based upon the observation that, in spite of the evolving nature of the heap, several of its properties remain stable. HeapMD uses this observation in a novel way: periodically, during the execution of the program, it computes a suite of metrics which are sensitive to the state of the heap. These metrics track heap behavior, and the stability of t ...

Keywords: anomaly detection, bugs, debugging, heap, metrics

7 Understanding the connectivity of heap objects
Martin Hirzel, Johannes Henkel, Amer Diwan, Michael Hind
June 2002 **ACM SIGPLAN Notices , Proceedings of the 3rd international symposium on Memory management ISMM '02**, Volume 38 Issue 2 supplement
Publisher: ACM Press
Full text available: pdf(256.15 KB) Additional Information: full citation, abstract, references, citings, index terms

Modern garbage collectors partition the set of heap objects to achieve the best performance. For example, generational garbage collectors partition objects by age and focus their efforts on the youngest objects. Partitioning by age works well for many programs because younger objects usually have short lifetimes and thus garbage collection of young objects is often able to free up many objects. However, generational garbage collectors are typically much less efficient for longer-lived objects, a ...

Keywords: connectivity based garbage collection, object lifetimes

8 Invited talk: Intermediate-representation recovery from low-level code
Thomas Reps, Gogul Balakrishnan, Junghee Lim
January 2006 **Proceedings of the 2006 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation PEPM '06**
Publisher: ACM Press
Full text available: pdf(301.12 KB) Additional Information: full citation, abstract, references, index terms

The goal of our work is to create tools that an analyst can use to understand the workings of COTS components, plugins, mobile code, and DLLs, as well as memory snapshots of worms and virus-infected code. This paper describes how static analysis provides techniques that can be used to recover intermediate representations that are similar to those that can be created for a program written in a high-level language.

9 Deriving object typestates in the presence of inter-object references
Mangala Gowri Nanda, Christian Grothoff, Satish Chandra
October 2005 **ACM SIGPLAN Notices , Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '05**, Volume 40 Issue 10
Publisher: ACM Press
Full text available: pdf(557.45 KB) Additional Information: full citation, abstract, references, index terms

We are interested in static analysis of Java classes with the goal of discovering the preconditions under which a certain program point within a method may be reached, taking into account the effects of previous method calls on an object of that class. The information pertinent to this computation is represented as the object's typestate, which is a finite set of relevant predicates that abstract the object's actual state. The execution of a method depends on an object's current typestate as wel ...

Keywords: Java, alias analysis, heap analysis, interface specification, predicate abstraction

10 Type-safe linking and modular assembly language

Neal Glew, Greg Morrisett
January 1999 **Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '99**
Publisher: ACM Press
Full text available: pdf(1.36 MB) Additional Information: full citation, references, citings, index terms

11 Compilers 2: Instance-wise points-to analysis for loop-based dependence testing
Peng Wu, Paul Feautrier, David Padua, Zehra Sura
June 2002 **Proceedings of the 16th international conference on Supercomputing ICS '02**
Publisher: ACM Press
Full text available: pdf(267.28 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a points-to analysis that aims at enabling loop-based dependence analysis in the presence of Java references. The analysis is based on an abstraction called *element-wise points-to* (ewpt) *mapping*. An ewpt mapping summarizes, in a compact representation, the relation between a pointer and the heap object it points to, for every *instance* of the pointer inside a loop and for every *array element* directly accessible through this pointer. Such instance-wise and el ...

Keywords: Java, dependence analysis, heap analysis, pointer analysis, pointer arrays

12 Conditional must not aliasing for static race detection
Mayur Naik, Alex Aiken
January 2007 **ACM SIGPLAN Notices , Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '07**, Volume 42 Issue 1
Publisher: ACM Press
Full text available: pdf(675.24 KB) Additional Information: full citation, abstract, references, index terms

Race detection algorithms for multi-threaded programs using the common lock-based synchronization idiom must correlate locks with the memory locations they guard. The heart of a proof of race freedom is showing that if two locks are distinct, then the memory locations they guard are also distinct. This is an example of a general property we call *conditional must not aliasing*: Under the assumption that two objects are not aliased, prove that two other objects are not aliased. This paper in ...

Keywords: Java, concurrency, multi-threading, static race detection, synchronization

13 Type-Safe linking with recursive DLLs and shared libraries
Dominic Duggan
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6
Publisher: ACM Press
Full text available: pdf(658.62 KB) Additional Information: full citation, abstract, references, citings, index terms

Component-based programming is an increasingly prevalent theme in software development, motivating the need for expressive and safe module interconnection languages. Dynamic linking is an important requirement for module interconnection languages, as exemplified by dynamic link libraries (DLLs) and class loaders in operating systems and Java, respectively. A semantics is given for a type-safe module interconnection language that supports shared libraries and dynamic linking, as well as circular ...

Keywords: Dynamic Linking, Module Interconnection Languages, Recursive Modules, Shared Libraries

14 Automatic pool allocation for disjoint data structures
Chris Lattner, Vikram Adve
June 2002 **ACM SIGPLAN Notices , Proceedings of the 2002 workshop on Memory system performance MSP '02**, Volume 38 Issue 2 supplement
Publisher: ACM Press
Full text available: pdf(1.48 MB) Additional Information: full citation, abstract, references, citings

This paper presents an analysis technique and a novel program transformation that can enable powerful optimizations for entire linked data structures. The fully automatic transformation converts ordinary programs to use pool (aka region) allocation for heap-based data structures. The transformation relies on an efficient link-time interprocedural analysis to identify disjoint data structures in the program, to check whether these data structures are accessed in a type-safe manner, and to constru ...

15 Cache-conscious data placement
Brad Calder, Chandra Krintz, Simmi John, Todd Austin
October 1998 **ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the eighth international conference on Architectural support for programming languages and operating systems ASPLOS-VIII**, Volume 33 , 32 Issue 11 , 5
Publisher: ACM Press
Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, citings, index terms

As the gap between memory and processor speeds continues to widen, cache eficiency is an increasingly important component of processor performance. Compiler techniques have been used to improve instruction cache pet$ormance by mapping code with temporal locality to different cache blocks in the virtual address space eliminating cache conflicts. These code placement techniques can be applied directly to the problem of placing data for improved data cache pedormance.In this paper we present a gene ...

16 Software prefetching for mark-sweep garbage collection: hardware analysis and software redesign
Chen-Yong Cher, Antony L. Hosking, T. N. Vijaykumar
October 2004 **ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , ACM SIGARCH Computer Architecture News , Proceedings of the 11th international conference on Architectural support for programming languages and operating systems ASPLOS-XI**, Volume 38 , 39 , 32 Issue 5 , 11 , 5
Publisher: ACM Press
Full text available: pdf(165.32 KB) Additional Information: full citation, abstract, references, citings, index terms

Tracing garbage collectors traverse references from live program variables, transitively tracing out the closure of live objects. Memory accesses incurred during tracing are essentially random: a given object may contain references to any other object. Since application heaps are typically much larger than hardware caches, tracing results in many cache misses. Technology trends will make cache misses more important, so tracing is a prime target for prefetching.Simulation of Java benchmarks runni ...

Keywords: breadth-first, buffered prefetch, cache architecture, depth-first, garbage collection, mark-sweep, prefetch-on-grey, prefetching

17 Garbarge collection for Prolog based on WAM
K. Appleby, M. Carllson, S. Haridi, D. Sawhlin
June 1988 **Communications of the ACM**, Volume 31 Issue 6
Publisher: ACM Press
Full text available: pdf(1.91 MB) Additional Information: full citation, abstract, references, citings, index terms

The Warren abstract machine (WAM) has become a generally accepted standard Prolog implementation technique. Garbage collection is an important aspect in the implementation of any Prolog system. A synopsis of the WAM is presented and then marking and compaction algorithms are shown that take advantage of WAM's unique use of the data areas. Marking and compaction are performed on both the heap and the trail; both use pointer reversal techniques, which obviate the need for extra stack space. H ...

18 Distributed garbage collection
J. D. Eckart, R. J. LeBlanc
July 1987 **ACM SIGPLAN Notices , Papers of the Symposium on Interpreters and interpretive techniques SIGPLAN '87**, Volume 22 Issue 7
Publisher: ACM Press
Full text available: pdf(684.69 KB) Additional Information: full citation, abstract, citings, index terms

There are two basic approachs to the problem of storage reclamation, process- and processor-based, named for the view point used to recognize when a particular piece of storage can be reclaimed. Examples of the processor approach include mark/sweep and copying algorithms and their variants, while reference counting schemes use a process view of the collection. It is argued that the process approach is better suited for distributed computation where links between dynamically allocated objects may ...

19 Energy-aware on-demand routing protocols for wireless ad hoc networks
Baoxian Zhang, Hussein T. Mouftah
July 2006 **Wireless Networks**, Volume 12 Issue 4
Publisher: Kluwer Academic Publishers
Full text available: pdf(387.74 KB) Additional Information: full citation, abstract, references, index terms

Energy use is a crucial design concern in wireless ad hoc networks since wireless terminals are typically battery-operated. The design objectives of energy-aware routing are two folds: Selecting energy-efficient paths and minimizing the protocol overhead incurred for acquiring such paths. To achieve these goals simultaneously, we present the design of several on-demand energy-aware routing protocols. The key idea behind our design is to adaptively select the subset of nodes that are required to ...

Keywords: energy use, routing, wireless ad hoc networks

20 The KaffeOS Java runtime system
Godmar Back, Wilson C. Hsieh
July 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 4
Publisher: ACM Press
Full text available: pdf(704.30 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Single-language runtime systems, in the form of Java virtual machines, are widely deployed platforms for executing untrusted mobile code. These runtimes provide some of the features that operating systems provide: interapplication memory protection and basic system services. They do not, however, provide the ability to isolate applications from each other. Neither do they provide the ability to limit the resource consumption of applications. Consequently, the performance of current systems degra ...

Keywords: Robustness, garbage collection, isolation, language runtimes, resource management, termination, virtual machines

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

memory leak based on links between objects

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used: memory leak based on links between objects

Found **136,389** of **207,474**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200
Best 200 shown

Result page: **1** 2 3 4 5 6 7 8 9 10 next

Relevance scale

1 Cryptography and data security
Dorothy Elizabeth Robling Denning
January 1982 Book
Publisher: Addison-Wesley Longman Publishing Co., Inc.
Full text available: pdf(19.47 MB) Additional Information: full citation, abstract, references, cited by, index terms

From the Preface (See Front Matter for full Preface)

Electronic computers have evolved from exiguous experimental enterprises in the 1940s to prolific practical data processing systems in the 1980s. As we have come to rely on these systems to process and store data, we have also come to wonder about their ability to protect valuable data.

Data security is the science and study of methods of protecting data in computer and communication systems from unauthorized disclosure ...

2 HeapMD: identifying heap-based bugs using anomaly detection
Trishul M. Chilimbi, Vinod Ganapathy
October 2006 **ACM SIGARCH Computer Architecture News , ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , Proceedings of the 12th international conference on Architectural support for programming languages and operating systems ASPLOS-XII**, Volume 34 , 40 , 41 Issue 5 , 5 , 11
Publisher: ACM Press
Full text available: pdf(246.89 KB) Additional Information: full citation, abstract, references, index terms

We present the design, implementation, and evaluation of HeapMD, a dynamic analysis tool that finds heap-based bugs using anomaly detection. HeapMD is based upon the observation that, in spite of the evolving nature of the heap, several of its properties remain stable. HeapMD uses this observation in a novel way: periodically, during the execution of the program, it computes a suite of metrics which are sensitive to the state of the heap. These metrics track heap behavior, and the stability of t ...

Keywords: anomaly detection, bugs, debugging, heap, metrics

3 A practical flow-sensitive and context-sensitive C and C++ memory leak detector

David L. Heine, Monica S. Lam
May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation PLDI '03**, Volume 38 Issue 5
Publisher: ACM Press
Full text available: pdf(214.44 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a static analysis tool that can automatically find memory leaks and deletions of dangling pointers in large C and C++ applications.We have developed a type system to formalize a practical ownership model of memory management. In this model, every object is pointed to by one and only one *owning* pointer, which holds the exclusive right and obligation to either delete the object or to transfer the right to another owning pointer. In addition, a pointer-typed class member ...

Keywords: error detection, memory leaks, memory management, program analysis, type systems

4 On the usefulness of type and liveness accuracy for garbage collection and leak detection
Martin Hirzel, Amer Diwan, Johannes Henkel
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6
Publisher: ACM Press
Full text available: pdf(684.85 KB) Additional Information: full citation, abstract, references, citings, index terms

The effectiveness of garbage collectors and leak detectors in identifying dead objects depends on the *accuracy* of their reachability traversal. Accuracy has two orthogonal dimensions: (i) whether the reachability traversal can distinguish between pointers and nonpointers (*type accuracy*), and (ii) whether the reachability traversal can identify memory locations that will be dereferenced in the future (*liveness accuracy*). This article presents an experimental study of the impo ...

Keywords: Conservative garbage collection, leak detection, liveness accuracy, program analysis, type accuracy

5 Evolving a language in and for the real world: C++ 1991-2006
Bjarne Stroustrup
June 2007 **Proceedings of the third ACM SIGPLAN conference on History of programming languages HOPL III**
Publisher: ACM Press
Full text available: pdf(838.10 KB) Additional Information: full citation, abstract, references, index terms

This paper outlines the history of the C++ programming language from the early days of its ISO standardization (1991), through the 1998 ISO standard, to the later stages of the C++0x revision of that standard (2006). The emphasis is on the ideals, constraints, programming techniques, and people that shaped the language, rather than the minutiae of language features. Among the major themes are the emergence of generic programming and the STL (the C++ standard library's algorithms and container ...

Keywords: C++, ISO, STL, evolution, history, language use, libraries, multi-paradigm programming, programming language, standardization

6 Nonblocking memory management support for dynamic-sized data structures

Maurice Herlihy, Victor Luchangco, Paul Martin, Mark Moir
May 2005 **ACM Transactions on Computer Systems (TOCS)**, Volume 23 Issue 2
Publisher: ACM Press
Full text available: pdf(944.89 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Conventional dynamic memory management methods interact poorly with lock-free synchronization. In this article, we introduce novel techniques that allow lock-free data structures to allocate and free memory dynamically using any thread-safe memory management library. Our mechanisms are lock-free in the sense that they do not allow a thread to be prevented from allocating or freeing memory by the failure or delay of other threads. We demonstrate the utility of these techniques by showing how to m ...

Keywords: Multiprocessors, concurrent data structures, dynamic data structures, memory management, nonblocking synchronization

7 The KaffeOS Java runtime system
Godmar Back, Wilson C. Hsieh
July 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 4
Publisher: ACM Press
Full text available: pdf(704.30 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Single-language runtime systems, in the form of Java virtual machines, are widely deployed platforms for executing untrusted mobile code. These runtimes provide some of the features that operating systems provide: interapplication memory protection and basic system services. They do not, however, provide the ability to isolate applications from each other. Neither do they provide the ability to limit the resource consumption of applications. Consequently, the performance of current systems degra ...

Keywords: Robustness, garbage collection, isolation, language runtimes, resource management, termination, virtual machines

8 The theory of parsing, translation, and compiling
Alfred V. Aho, Jeffrey D. Ullman
January 1972 Book
Publisher: Prentice-Hall, Inc.
Full text available: pdf(98.28 MB) Additional Information: full citation, abstract, references, cited by, index terms

From volume 1 Preface (See Front Matter for full Preface)

This book is intended for a one or two semester course in compiling theory at the senior or graduate level. It is a theoretically oriented treatment of a practical subject. Our motivation for making it so is threefold.

(1) In an area as rapidly changing as Computer Science, sound pedagogy demands that courses emphasize ideas, rather than implementation details. It is our hope that the algorithms and concepts presen ...

9 Context- and path-sensitive memory leak detection
Yichen Xie, Alex Aiken
September 2005 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 10th European software engineering conference held jointly with 13th**

ACM SIGSOFT international symposium on Foundations of software engineering ESEC/FSE-13, Volume 30 Issue 5
Publisher: ACM Press
Full text available: pdf(251.65 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a context- and path-sensitive algorithm for detecting memory leaks in programs with explicit memory management. Our leak detection algorithm is based on an underlying escape analysis: any allocated location in a procedure *P* that is not deallocated in *P* and does not escape from *P* is leaked. We achieve very precise context- and path-sensitivity by expressing our analysis using boolean constraints. In experiments with six large open source projects our analysis produ ...

Keywords: boolean satisfiability, error detection, memory leaks, memory management, program analysis

10 Understanding memory allocation of scheme programs
Manuel Serrano, Hans-J. Boehm
September 2000 **ACM SIGPLAN Notices , Proceedings of the fifth ACM SIGPLAN international conference on Functional programming ICFP '00**, Volume 35 Issue 9
Publisher: ACM Press
Full text available: pdf(821.49 KB) Additional Information: full citation, abstract, references, citings, index terms

Memory is the performance bottleneck of modern architectures. Keeping memory consumption as low as possible enables fast and unobtrusive applications. But it is not easy to estimate the memory use of programs implemented in functional languages, due to both the complex translations of some high level constructs, and the use of automatic memory managers.To help understand memory allocation behavior of Scheme programs, we have designed two complementary tools. The first one reports on frequency of ...

11 Artificial intelligence
Elaine Rich
January 1983 Book
Publisher: McGraw-Hill, Inc.
Additional Information: full citation, abstract, references, cited by, review

The goal of this book is to provide programmers and computer scientists with a readable introduction to the problems and techniques of artificial intelligence (A.I.). The book can be used either as a text for a course on A.I. or as a self-study guide for computer professionals who want to learn what A.I. is all about.

The book was designed as the text for a one-semester, introductory graduate course in A.I. In such a course, it should be possible to cover all of the material in the boo ...

12 Memory allocation: Gated memory control for memory monitoring, leak detection and garbage collection
Chen Ding, Chengliang Zhang, Xipeng Shen, Mitsunori Ogihara
June 2005 **Proceedings of the 2005 workshop on Memory system performance MSP '05**
Publisher: ACM Press
Full text available: pdf(135.93 KB) Additional Information: full citation, abstract, references, citings, index terms

In the past, program monitoring often operates at the code level, performing checks at function and loop boundaries. Recent research shows that profiling analysis can identify high-level phases in complex binary code. Examples are time steps in scientific simulations and service cycles in utility programs. Because of their larger size and more predictable behavior, program phases make it possible for more accurate and longer term predictions of program behavior, especially its memory usage. This ...

Keywords: memory leak, memory usage monitoring, object life, preventive memory management, program phase

13 Cork: dynamic memory leak detection for garbage-collected languages
Maria Jump, Kathryn S. McKinley
January 2007 **ACM SIGPLAN Notices , Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '07**, Volume 42 Issue 1
Publisher: ACM Press
Full text available: pdf(651.25 KB) Additional Information: full citation, abstract, references, index terms

A *memory leak* in a garbage-collected program occurs when the program inadvertently maintains references to objects that it no longer needs. Memory leaks cause systematic heap growth, degrading performance and resulting in program crashes after perhaps days or weeks of execution. Prior approaches for detecting memory leaks rely on heap differencing or detailed object statistics which store state proportional to the number of objects in the heap. These overheads preclude their use on the sa ...

Keywords: dynamic, garbage collection, memory leak detection, memory leaks, runtime analysis

14 Understanding the connectivity of heap objects
Martin Hirzel, Johannes Henkel, Amer Diwan, Michael Hind
June 2002 **ACM SIGPLAN Notices , Proceedings of the 3rd international symposium on Memory management ISMM '02**, Volume 38 Issue 2 supplement
Publisher: ACM Press
Full text available: pdf(256.15 KB) Additional Information: full citation, abstract, references, citings, index terms

Modern garbage collectors partition the set of heap objects to achieve the best performance. For example, generational garbage collectors partition objects by age and focus their efforts on the youngest objects. Partitioning by age works well for many programs because younger objects usually have short lifetimes and thus garbage collection of young objects is often able to free up many objects. However, generational garbage collectors are typically much less efficient for longer-lived objects, a ...

Keywords: connectivity based garbage collection, object lifetimes

15 Level set and PDE methods for computer graphics
David Breen, Ron Fedkiw, Ken Museth, Stanley Osher, Guillermo Sapiro, Ross Whitaker
August 2004 **ACM SIGGRAPH 2004 Course Notes SIGGRAPH '04**
Publisher: ACM Press
Full text available: pdf(17.07 MB) Additional Information: full citation, abstract, citings

Level set methods, an important class of partial differential equation (PDE) methods, define dynamic surfaces implicitly as the level set (iso-surface) of a sampled, evolving nD function. The course begins with preparatory material that introduces the concept of using partial differential equations to solve problems in computer graphics, geometric modeling and computer vision. This will include the structure and behavior of several different types of differential equations, e.g. the level set eq ...

16 Charles W. Bachman interview: September 25-26, 2004; Tucson, Arizona
Thomas Haigh
January 2006 **ACM Oral History interviews**
Publisher: ACM Press
Full text available: pdf(761.66 KB) Additional Information: full citation, abstract

Charles W. Bachman reviews his career. Born during 1924 in Kansas, Bachman attended high school in East Lansing, Michigan before joining the Army Anti Aircraft Artillery Corp, with which he spent two years in the Southwest Pacific Theater, during World War II. After his discharge from the military, Bachman earned a B.Sc. in Mechanical Engineering in 1948, followed immediately by an M.Sc. in the same discipline, from the University of Pennsylvania. On graduation, he went to work for Do ...

17 Formalizing the safety of Java, the Java virtual machine, and Java card
Pieter H. Hartel, Luc Moreau
December 2001 **ACM Computing Surveys (CSUR)**, Volume 33 Issue 4
Publisher: ACM Press
Full text available: pdf(442.86 KB) Additional Information: full citation, abstract, references, citings, index terms

We review the existing literature on Java safety, emphasizing formal approaches, and the impact of Java safety on small footprint devices such as smartcards. The conclusion is that although a lot of good work has been done, a more concerted effort is needed to build a coherent set of machine-readable formal models of the whole of Java and its implementation. This is a formidable task but we believe it is essential to build trust in Java safety, and thence to achieve ITSEC level 6 or Common Crite ...

Keywords: Common criteria, programming

18 Memory safety without garbage collection for embedded applications
Dinakar Dhurjati, Sumant Kowshik, Vikram Adve, Chris Lattner
February 2005 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 4 Issue 1
Publisher: ACM Press
Full text available: pdf(511.25 KB) Additional Information: full citation, abstract, references, citings, index terms

Traditional approaches to enforcing memory safety of programs rely heavily on run-time checks of memory accesses and on garbage collection, both of which are unattractive for embedded applications. The goal of our work is to develop advanced compiler techniques for enforcing memory safety with minimal run-time overheads. In this paper, we describe a set of compiler techniques that, together with minor semantic restrictions on C programs and no new syntax, ensure memory safety and provide most of ...

Keywords: Embedded systems, automatic pool allocation, compilers, programming languages, region management, security, static analysis

19 Saturn: A scalable framework for error detection using Boolean satisfiability
Yichen Xie, Alex Aiken
May 2007 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 29 Issue 3
Publisher: ACM Press
Full text available: pdf(742.22 KB) Additional Information: full citation, abstract, references, index terms

This article presents Saturn, a general framework for building precise and scalable static error detection systems. Saturn exploits recent advances in Boolean satisfiability (SAT) solvers and is path sensitive, precise down to the bit level, and models pointers and heap data. Our approach is also highly scalable, which we achieve using two techniques. First, for each program function, several optimizations compress the size of the Boolean formulas that model the control flow and data flow and ...

Keywords: Boolean satisfiability, Program analysis, error detection

20 Link and channel measurement: A simple mechanism for capturing and replaying wireless channels
Glenn Judd, Peter Steenkiste
August 2005 **Proceeding of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis E-WIND '05**
Publisher: ACM Press
Full text available: pdf(6.06 MB) Additional Information: full citation, abstract, references, index terms

Physical layer wireless network emulation has the potential to be a powerful experimental tool. An important challenge in physical emulation, and traditional simulation, is to accurately model the wireless channel. In this paper we examine the possibility of using on-card signal strength measurements to capture wireless channel traces. A key advantage of this approach is the simplicity and ubiquity with which these measurements can be obtained since virtually all wireless devices provide the req ...

Keywords: channel capture, emulation, wireless

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next